The Right Question

How to stop worrying about AI and start running a team of bots

---

The short version

You will not fall behind because AI got smarter than you. You will fall behind if you can't tell it what you want.

That is the whole argument of this guide. The rest is how to get good at it: what a good job for an AI looks like, how to tell a vague ask from a useful one, where to start in your own field, and a seven-day plan to try it on real work.

Where this comes from

In a September 2026 interview with China Media Group (published on CGTN's YouTube channel on 25 September 2026), Elon Musk was asked how a 20-year-old can avoid being left behind during the AI transition. His answer, from about 23:12 in the video:

> "My recommendation for education would be to get as broad-based education as possible. That includes arts and sciences, engineering, wide general knowledge. In order to know what to ask the robots for, you need to be able to formulate the question. And with the broadest possible general knowledge, you'll be better able to formulate the questions, or what they call prompt engineering, I guess. You've got to convey your wishes to the robot."

Source: *Full: Exclusive CMG interview with Tesla CEO Elon Musk*, CGTN, YouTube, https://www.youtube.com/watch?v=34-5wOGmCKc

He then predicts a world with so much AI that it will be "eager to hear any request that you could possibly have, because that request will be fulfilled immediately." That part is a forecast, and you

can take it or leave it. The first part is advice you can use today, whether or not the forecast comes true.

The thesis

Here is the chain, one link at a time:

Broad knowledge -> the right question -> a team of bots -> done.

1. **Broad knowledge.** You can only ask for what you can imagine. If you know a little about design, a little about law, a little about statistics, you notice more problems and more ways to solve them. You also notice when an answer is wrong.
2. **The right question.** Knowledge alone does nothing. You have to turn it into a clear job: what you want, why, under which limits, and what "done" looks like.
3. **A team of bots.** Once you can write one clear job, you can write five. Agents like Grok Bot can take a job, work through it with tools, and report back. Several of them can run in parallel on different parts of a problem.
4. **Done.** The work comes back. You review it, correct it, and send the next job.

The bottleneck moves. It used to be "can I do this work?" Now it is "can I describe this work well enough that someone else, or something else, can do it?" That is a management skill, and most people have never been taught it.

My opinion, plainly: most people will wait for someone to tell them what to learn next. A smaller group will practice giving AI clear jobs, and they will end up with real leverage. That is not a statistic. It is a bet. This guide is for people who want to be in the second group.

The framework: five parts of a good job

Every good job you give an AI has five parts. You can write them in a few sentences. Skip one and you will usually get back something generic, wrong, or unusable.

1. Goal

What do you want, and why? The "why" matters more than people think. "Write a cold email" and "write a cold email that gets a 15-minute call with a hospital procurement lead who has never heard of us" produce very different work.

Test: could a smart stranger tell what a great result looks like from your goal alone?

2. Context

What does the AI need to know that it can't guess? Who you are, who the audience is, what you have already tried, what files or links are relevant, what the tone should be.

Test: if you handed this to a new hire on day one, what would they ask you first? Answer that in advance.

3. Constraints

The limits. Budget, length, deadline, tools allowed, things it must not do. "Do not contact anyone." "Under 200 words." "Only use sources from the last 12 months." "Do not change the database."

Test: what would make you angry if it happened? Write that down as a constraint.

4. Success criteria

How you will judge the result. Be concrete. "Three options, each with a price and a source link." "Code that passes the existing tests." "A table I can paste into a spreadsheet."

Test: could you check the result against your criteria in under five minutes?

5. What to report back

What you want to see at the end, and in what shape. A summary, a file, a list of open questions, what it could not verify. Asking it to tell you what it is unsure about is the single cheapest way to catch mistakes.

Test: if you only read the first three lines of the report, would you know whether to trust it?

A short way to remember it: **Goal, Context, Constraints, Criteria, Report.**

Worked examples: bad ask vs good ask

Example 1: a student picking a thesis topic

Bad ask:

> Give me thesis ideas about AI.

You will get ten generic topics that thousands of other students are also getting.

Good ask:

> **Goal:** I need a master's thesis topic in economics that I can finish in eight months and that would interest a fintech employer. **Context:** I'm at a university in Bratislava. I know Python and basic econometrics. My supervisor works on labour markets. I have access to Eurostat and Slovak statistics office data. **Constraints:** Only topics that can be done with public data. No topics that need a lab or surveys of more than 100 people. **Success criteria:** Five topics. For each: the research question in one sentence, the dataset, the method, and one recent paper (with link) that is close but not the same. **Report back:** Rank them by how feasible they are in eight months, and flag any where you could not find the dataset.

Same student, same tool. The second ask produces something you can take to your supervisor.

Example 2: a founder researching competitors

Bad ask:

> Research my competitors.

Good ask:

> **Goal:** I want to understand how three competitors price and position their product, so I can decide our pricing for the launch next month. **Context:** We sell scheduling software to dental clinics in Central Europe. Competitors: [A], [B], [C]. Our current draft price is EUR 49 per clinic per month. **Constraints:** Public information only. Do not sign up for trials or contact anyone. **Success criteria:** A table with plan names, prices, what is included, and the main claim on each homepage. A link for every number. **Report back:** Three sentences on where we are cheaper or more expensive, and anything you could not find.

Note the last line. You are asking it to admit gaps instead of filling them with guesses.

Example 3: an operator clearing a support backlog

Bad ask:

> Help me with support tickets.

Good ask:

> **Goal:** Clear the oldest 40 tickets in our support inbox this afternoon. **Context:** Our refund policy is in the attached doc. Most old tickets are about delayed shipping. **Constraints:** Draft replies only. Do not send anything. Do not promise refunds over EUR 50 without

> flagging them to me. **Success criteria:** Every ticket has either a draft reply or a note saying why it needs me. **Report back:** A list grouped into "ready to send", "needs a decision", and "needs more info", with the ticket IDs.

The pattern in all three: the good ask is longer, and it saves time. Three minutes writing it saves an hour of back and forth.

Going from one bot to a team

Once you can write one good job, the next step is splitting a bigger goal into several jobs that can run at the same time.

A simple rule: **split by output, not by step.** Don't make one bot do research, then another write, then another edit, all waiting on each other. Give each bot its own finished piece. One researches competitor A, one researches competitor B, one drafts the pricing page. You review and combine.

Three habits that keep a team of bots useful:

- **One owner per job.** Each job has one bot and one clear deliverable. No overlaps.
- **Shared context in one place.** Keep a short brief (who you are, what the project is, the rules) and paste it into every job, or keep it in a file every bot can read.
- **You stay the editor.** The bots do the work. You decide what is good enough. Reviewing is your job, and it is where your broad knowledge pays off.

Pick your field

These are free guides from SpaceXAI on using Grok Bot for specific kinds of work. Each link was checked on 11 November 2026 and loaded. Start with the first one, then pick the one closest to your work.

- **Getting started** - for anyone who has never handed real work to a bot. https://x.ai/bot/guides/getting-started
- **Engineering** - for developers who want several engineer bots working on code in parallel. https://x.ai/bot/guides/grok-bot-for-engineering
- **Mobile app development** - for solo builders or small teams making an app or game with a set of bots. https://x.ai/bot/guides/grok-bot-for-mobile-app-development

- **Design -** for designers who want to explore more ideas than they have time to sketch. https://x.ai/bot/guides/designing-grok-bot-with-grok-bot
- **Product management -** for PMs who want research, specs, and follow-ups handled by a team that reports to them. https://x.ai/bot/guides/grok-bot-for-pms
- **Marketing -** for marketers researching competitors and drafting positioning. https://x.ai/bot/guides/grok-bot-for-marketing
- **Founder-led sales -** for founders doing their own prospecting, research, and outreach drafts. https://x.ai/bot/guides/grok-bot-for-founder-led-sales
- **Recruiting -** for hiring managers and recruiters doing interview prep, scorecards, and sourcing. https://x.ai/bot/guides/grok-bot-for-recruiting
- **Legal ops -** for in-house legal and ops people triaging contracts and preparing for meetings. https://x.ai/bot/guides/grok-bot-for-legal-ops
- **Support -** for support leads who want to work through the whole queue with bots. https://x.ai/bot/guides/grok-bot-for-support
- **Templates -** for anyone who built a bot setup that works and wants to share it as a reusable recipe. https://x.ai/bot/guides/templates-for-grok-bot

Read one guide fully before skimming the others. Depth beats coverage here.

Your 7-day starter plan

Thirty to sixty minutes a day. Use real work, not toy examples. The point is to build the habit of writing clear jobs.

Day 1 - Write down your backlog. List ten tasks you have been putting off or that eat your week. Next to each, write one sentence: what would "done" look like?

Day 2 - One job, five parts. Pick the easiest task. Write it out with Goal, Context, Constraints, Criteria, Report. Give it to a bot. Compare what came back to your criteria. Note what was missing from your ask.

Day 3 - Rewrite and rerun. Take yesterday's job, fix the ask based on what went wrong, and run it again. This is the most important day. Most of the skill is in the rewrite.

Day 4 - Read your field's guide. Pick the guide above that fits your work. Read it fully. Write down one setup from it you want to copy.

Day 5 - Two bots at once. Pick a task that splits into two independent pieces. Write two jobs and run them in parallel. Combine the results yourself.

Day 6 - Learn something outside your field. Spend 30 minutes on a subject next to yours: a designer reads about pricing, an engineer reads about sales, a student reads about contract basics. Then write one job that uses what you learned. This is the broad-knowledge part of the chain, and it compounds.

Day 7 - Save your best job as a template. Take the job that worked best this week and save it as a reusable template, with blanks for the parts that change. Next week, start from the template instead of a blank page.

After seven days you will have a backlog, a few templates, and a clear sense of where your asks are weak. That is more than most people have.

The mistakes to avoid

- **Asking for "ideas" when you need a decision.** Ask for options with trade-offs and a recommendation.
- **Hiding the real goal.** If you want a job offer, say so. If you want to cut costs, say by how much.
- **Not saying what it must not do.** Bots will act on what you allow. State the limits.
- **Trusting without checking.** Ask what it could not verify. Check links and numbers before you use them.
- **Starting over every time.** Save good jobs as templates. Your library of asks is an asset.

The point

AI will do more of the work every year. That part is not up to you. What is up to you is whether you are the person who knows what to ask for, can tell a good result from a bad one, and can keep a team of bots busy on things that matter.

That skill has two halves: knowing enough about the world to ask good questions, and knowing how to turn a question into a clear job. The first takes years and never really ends. The second you can start this week.

[CTA: join / get the full product]